# Introduction

## Overview

Hierarchical Internet of Things (HIOT) is a communication protocol for the Internet of Things (IOT). It is designed to provide a full suite of messaging for the configuration, management and operational functionality of any IOT use case.

In addition to traditional IOT messaging, HIOT includes message types which are specifically designed to support delegation of control as well as data aggregation and processing at or near the source of the data. This delegation and local data processing is where HIOT gets its name, enabling IOT to move from a traditional flat structure to a hierarchy of devices and associated roles in order to deliver maximum scalability.

There is also a specific message type aimed at increased interoperability. Handler messages, which allow new functionality to be added to any remote device at any time, can be used to integrate with third party components and translate between HIOT and protocols.

During the development of the protocol, a particular use case was selected in order to test the suitability of the specific message types, a “smart” office building. To examine the practical functionality, a partial implementation of the protocol was developed in NodeJS with MQTT as the messaging protocol.

While this document makes numerous references to MQTT throughout, there should no reason why other messaging protocols could not be used in its place.

## Terminology

HIOT in intended to provide full scope communications for any IOT use-cases. In order to illustrate functionality, a number of terms are used to describe specific components. These terms are defined below.

### Platform

Platform refers to typical server-side components offered as part of a SaaS or cloud solution for IOT.

### Deployment

In our selected use case, a smart office building, the term deployment refers to all IOT elements contained within the building.

### Message

Work to date on HIOT has been based on MQTT and a message is any communication sent on an MQTT topic. However, this should be considered non-normative and there should be no reason why other protocols could be used in place of, or in addition to, MQTT.

### Network

In the context of HIOT, the network refers to the underlying network for the messaging protocol. Like MQTT, there is no apparent limitation in terms of what network protocols are used.

### Topic

The term topic is taken from the MQTT specification and is used in that context throughout this document. Channels and Paths for other messaging protocols should be equally applicable and HIOT should be fully implementable using any alternative to MQTT.

### Handler

Handlers are a combination of configuration settings and executables or scripts which can provide additional functionality within a deployment. In our partial NodeJS implementation, handler files are JavaScript files which are Base64 encoded as part of Handler messages sent to specific devices. These handler files could also be executable files, byte-code or any other type of files capable of being executed on a device.

### Cloud-side

“Cloud-side” refers to typical SaaS offerings, including remote administration tools and user-facing components. The cloud-side has been divided into a number of subcomponents offering specific functionality. While some of these components and their functionality is outside of the scope of HIOT, they are included or referenced here for the sake of completeness.

#### Platform, UI and Coordinator Link

Strictly speaking, the Coordinator Link is the only cloud-side element within the scope of HIOT. The Platform component refers to client or user facing tools, typically exposed through REST web-services and web-sockets. UI refers to any user interface connected to the Platform component.

The Coordinator Link (in conjunction with the deployment-side Coordinator, described below) is responsible for all communications between the Platform and a deployment.

Throughout the document, unless called out at the time, the term Platform refers to all cloud-side elements within IOT.

### Path

Paths for part of each topic sent and received by a deployment. For example, a health message would take the form “h/<device path>”, where <device path> is the unique path for the associated device.

Each role deployed to a device also has a unique path. For example, a physical device may have multiple sensors and/or controllers attached, with each one having its own unique path. This facilitates the movement of roles across devices without the need for widespread configuration change.

### Deployment-side

“Deployment-side” refers to the local IOT implementation - for example, our smart building or a production line in a factory. Within the Deployment-side, generic devices are configured via messages sent from the platform and their functionality is determined by their configured roles and associated handler files

#### Device

This is the base implementation for any component within a deployment. A device is capable of receiving configuration data from the platform and can transfer Health messages (described later) and Error messages to the platform.

#### Role: Actuator/controller

Controllers are responsible for interfacing with real world elements of the deployment such as heating and lighting controls etc. Handler files are deployed to devices to implement the required functionality, while the configuration message sent from the platform defines the required parameters, including a list of all commands and their parameters.

A device can have any number of configured controllers, with each controller being assigned its own messaging topic.

#### Role: Aggregator

Aggregators are a key component in HIOT and allow the deployment to process data locally rather than relying on cloud-side resources. In our selected use case, a number of temperature sensors, distributed across an open-plan area, can transmit readings to an aggregator which can calculate the mean, max and min temperature for the overall area. This data can then be fed to another aggregator which calculates the temperature across the entire floor. Based on the configuration and the handler file, raw sensor data can be included or omitted from the data transferred cloud-side.

#### Role: Broker

Brokers are organised in a hierarchical structure and pass encapsulated messages along to the relevant device or role. A broker topic is included in each message and defines the path to follow from the coordinator to the recipient device or role.

Similar to Aggregators, Brokers are designed to offer maximum scalability across a deployment by having commands and other platform messages chained across multiple levels.

#### Role: Coordinator

The Coordinator is the only deployment-side device with external connectivity. Each deployment has only one active Coordinator at any time. All messages between the cloud-side and deployment-side pass through the Coordinator, allowing for targeted hardening and security measures and decreasing the deployment’s attack surface.

#### Role: Sensor

Sensors provide real-world data to the deployment and can be integrated with a specified device through the use of configuration data from the Platform and an associated handler files.

A device can have any number of configured sensors, with each sensor having its own messaging topic.

#### Role: Commander

A commander is a local user-interface for the deployment. Unlike the UI component specified as part of the cloud-side, a Commander can communicate directly with specific aggregators and brokers via their messaging topics and subscriptions.

In our selected use-case, a Commander would likely take the form of a wall mounted touchscreen to control and monitor the environment in the immediate vicinity, e.g. control and monitor the temperature in lighting in an open plan office or manage conferencing equipment in a meeting room.

#### Example Implementation

The diagram below illustrates a minimal HIOT implementation.

D: A HIOT Device

S: Sensor – connected directly to a device or integrated via MQTT or another protocol. Configuration data from the Platform and a handler file are used to integrate the real-world sensor into HIOT

C: Controllers. Similar to Sensors, these can be directly connected or external and the use of handler files is intended to improve integration and interoperability.

A: Aggregators. Configuration data manages the topic subscriptions (from sensors or other aggregators) and publications (towards the cloud-side). Handler files are deployed to devices to perform operations on the data (e.g. read data from multiple sensors via subscriptions and send the mean, min and max of the reading to the platform via publications). For example, the Sensors connected to the aggregators in the diagram could all be temperature sensors - the first aggregator could compute mean, min and max temperatures for an open plan office, based on the readings from relevant sensors and subsequent aggregators could calculate the average temperature for a floor/building etc.

B: Brokers act as intermediaries between the platform and the controllers - so the broker in the diagram could control all the lighting/heating controls for the open plan office.

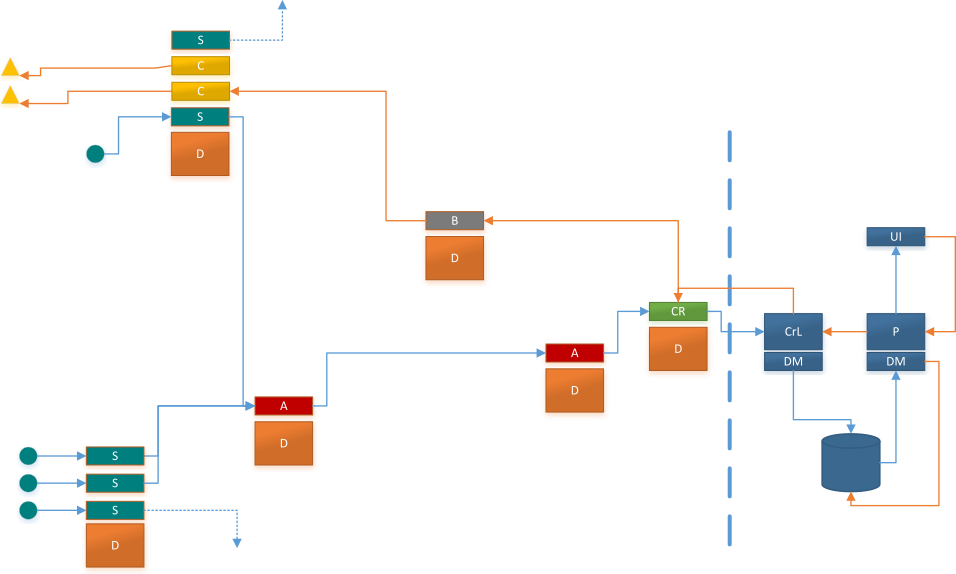
CR: Coordinator - essentially this is the gateway device for a deployment – a single point in and out of the deployment and the only local device which can communicate with the platform. In the event of a failure, this role can be moved to another device through the use of Coordinator messages across the deployment.

CrL: Coordinator link - this sits between the core platform and the deployment and coordinates messages between deployment-side and cloud-side.

DM: This is just a shared data layer used by both the cloud-side Platform and the Coordinator Link. Not relevant to HIOT, but included for completeness.

UI: This is any user facing solution connected to the Platform component.

All of the roles (and other device attributes) are controlled by device-specific configuration files which are managed cloud-side.



# Message Structure

In HIOT, MQTT topics are restricted to single character, base 58 elements. This is to reduce the size of the topic length and reduce the overall message size. Topic wildcards are not supported as part of HIOT. Base58 is used in favour of Base64 in order to improve readability and remove similar looking (0 and O etc.) from the topics.

The first character of the topic string represents the category of message being sent (discussed later in the document) and also the “direction of the message” – capital letters are used to denote a message from the platform to the deployment, while lower case letters are used for messages travelling in the opposite direction.

Topics are managed “platform-side” and it is the responsibility of the platform to ensure that each device and role within a particular deployment has a unique topic associated with it.

The list below highlights the number of distinct topics for a particular topic length, for example “?/a/b/c/d “, where “?” is the type message, for example, a health message from a device could use the following topic "h/1/2/5/B"

* 2 levels (“?/a”) = 58 devices and roles
* 3 levels (“?/a/b”) = 3,364 devices and roles
* 4 levels (“?/a/b/c”) = 195,112 devices and roles
* 5 levels (“?/a/b/c /d”) = 11,316,496 devices and roles

=> a 9-character topic string can support over 10,000,000 devices and roles in a single deployment.

As the platform is responsible for maintaining the configuration of each device and each role, deployment side components do not need to be aware of the relevant chains and paths required to transmit messages across the deployment.

Broker messages will always include the chain of brokers (in the form of a topic) and the destination address (topic) for the relevant device or role, and Aggregator configuration will include the topics for the relevant subscriptions and publications - this way devices do not need to maintain internal tables of paths etc. and roles can be moved across devices without the need to update the paths on multiple devices across the deployment.

# Control messages

## Coordinator sync: Z Topic

Coordinator sync message are sent between active and "hot-swap" Coordinators, and published from or subscribed to by other devices or the platform. These message are designed to ensure that there is one, and only one, active coordinator in the deployment.

Capital letters (“Z”) are used in messages from the active Coordinator, while any “hot-swap” or passive Coordinators will send message on the lower case topic (“z”).

Coordinator message topics are different from most in that there are no additional elements within the within the topic path. All coordinators subscribe to the coordinator topics (“Z” and “z”) and can publish on “z”, but only the currently active Coordinator can publish on (“Z”)

This has not been fully considered to date, but would expect to re-use concepts from other active-passive architecture

## On-boarding: O Topic

HIOT devices which do not have an active configuration, e.g. a new device which has just been connected to the deployment’s network, will publish an On-boarding message on topic “o”, containing a unique identifier for that device (typically the MAC address of the active network interface).

For example:

Topic: “o”,

Message: {

“deviceID”:””00-01-d3-33-76-9b”

}

All Aggregators subscribe to the On-boarding channel (“o”) and upon receiving an On-boarding message will request a new device path from the Platform by publishing on “o/<agg-path>” where, <agg-path> is the device ID of the aggregator.

The Platform then publishes the new, unique Device ID on topic “O/<agg-path>”. On receipt, of the message, the aggregator forwards it on topic “o/<deviceID>”. Duplicate On-boarding message for the same device, i.e. from other Aggregators, are ignored.



The new device then updates its configuration with the device path provided by the platform.

|  |  |  |
| --- | --- | --- |
| Message sender and Purpose | Path | Data |
| Device: Get a unique device path | “o” | Unique device ID : String |
| Aggregator: Request a new device id | “o/<first agg path>” | Unique device ID: String |
| Platform: set device path | “O/<first agg path>” | Unique device ID: String  Unique device Path: String |
| Aggregator: set device path | “O/<device id>” | Unique device Path: String |

Once the device path has been set, the device can be managed and configured from the platform.

## Device Config: C Topic

Device configuration is managed via the Platform through the use of message on the “C/<path>” and “c</path>” topics. A new configuration can be pushed to a specific device from the platform by pubishing on “C/<device path>”, where <device path> is the unique path for the relevant device (e.g. a device which has just been on-boarded).

Once the device has applied the relevant configuration, it responds to the platform on topic “c/<device path>”, with the updated configuration data. In the event that the platform has tried to configure an unsupported function (e.g. if the device is not fully HIOT compatible, or does not support all roles), the unsupported functions will be omitted from the device’s response.

In both cases, the message structure is the same.

Topic: “C/<device path>” or “c/<device path>”

Message: {

<configuration data as specified below>

}

In order to query the configuration of a specific device, an empty configuration message can be sent from the Platform, the device will then respond with its configuration data.

Each configuration message contains the entire configuration data for the specified device, including any roles deployed to the device.

|  |  |  |
| --- | --- | --- |
| Area | Element | Purpose |
| Device | hiotId | A unique identifier for the device, set by the Platform |
|  | deviceId | A unique identifier for the device, used as part of on-boarding (e.g. where no configuration data exists). Examples include MAC address or chip/board serial number |
|  | name | User friendly name for the device |
|  | description | A description of the device |
|  | devicePath | Unique path for the device topics |
| Device->MQTTServers | Configuration data for any MQTT brokers that the device connects to | |
| Roles->Broker | An array of zero or more broker implementations | |
|  | \_id | Unique ID for the particular broker implementation |
|  | path | Unique path for this implementation |
|  | deployment | Deployment ID - part of orphaned devices, discussed later |
|  | description | User friendly description of what the broker does |
|  | name | User friendly name for the implementation |
|  | handler | Handler file for required functionality |
|  | active | Boolean – is the broker implementation active? |
| Role->Coordinator | m2mMqttport | Port number for the Coordinator Link on the Platform |
|  | m2mMqttServer | IP Address or DNS name of the Coordinator Link |
|  | active | Boolean - Is this the active Coordinator? |
|  | auth | Object defining the credentials for the Coordinator Link |
| Role->Controller | An array of zero or more controllers installed on the device | |
|  | controllerId | Unique ID for this controller |
|  | description | User friendly description for the controller |
|  | name | User friendly name for the controller |
|  | deployment | Deployment ID - part of orphaned devices, discussed later |
|  | channel | Unique topic for this controller |
|  | handler | Handler file to implement required functionality/integration |
| Role->Controller->Commands | An array of commands for this controller | |
|  | c | Unique id for the command, usually just an incrementing int |
|  | n | User friendly name for the command |
|  | E | Command to execute |
|  | p | Array of parameters |
|  | p->p | ID of the parameter |
|  | p->datatype | Type name of the expected datatype |
|  | p->min | Min value (numeric), length (text) or size (other) for the parameter |
|  | p->max | Max value, length or size for the parameter |
|  | p->required | Boolean, is this a required parameter |
| Role->Aggregator | An array of zero or more aggregators installed on the device | |
|  | \_id | Unique ID for the aggregator |
|  | handler | Handler file to implement data processing/manipulation |
|  | name | User friendly name for the aggregator |
|  | channel | Unique channel to publish aggregated data on |
|  | description | User friendly description for the aggregator |
|  | poll | Frequency in milliseconds to generate and publish aggregated data |
|  | deployment | Deployment ID - part of orphaned devices, discussed later |
|  | Active | Is this aggregator active? |
|  | Include\_raw | Should the inbound data be included in the published data |
|  | topics | Array of topics to subscribe to for incoming data |
| Role->Sensor | Array of sensors to integrate with and publish data from | |
|  | \_id | Unique identifier for the sensor |
|  | name | User friendly name for the sensor |
|  | description | User friendly description for the sensor |
|  | handler | Handler file, for integration purposes |
|  | channel | Channel the sensor publishes on |
|  | Poll |  |
|  | config | Object containing configuration data for the specific sensor (handler) |
|  | location | Unique id of the relevant location in the deployment |
|  | active | Is this sensor active? |

Below is a typical device configuration message, with comments to aid interpretation

{

"device": {

"hiotId": “”, //String, the unique device ID set by the Platform

“deviceId”: “”, //String, unique ID of the device, e.g. MAC address

"name": “”, //String, user friendly name, assigned via platform

"description": "", //String, description, assigned via platform

"devicePath": "" //String, unique path for the device

},

"roles": { //array of installed roles on the device

"broker": [ //array of installed brokers

{

"\_id": "", //unique id for this broker

“path”:””, // topic to subscribe to

"deployment": "", // unique id for the relevant deployment

"description": "", // a description of what the broker does

"name": "", // name for the broker

"handler": "", // unique ID for the associated handler file

"active": true

}],

"coordinator": { // is this device a coordinator?

"m2mMqttport": 0, // Coordinator Link port number

"m2mMqttServer": "", // Coordinator Link address/name

“active”: true,

“auth”: {} // relevant authentication parameters for the platform

},

"controller": [{ //array of installed controllers

"\_id": "59c393ee54674b1ce4982a69",

"description": "Lighting controller - First Floor",

"name": "LWRF - F1",

"\_\_v": 0,

"deployment": "597f3056ef66be0648ef5bd3",

"controllerId": "cont1203", //unique name for this controller

"channel": "X/1/1/f1", //channel the controller accepts commands on

"handler": "59bfda6ed7649a3070551ad9",

"commands": {

"1": {

"name": "name of the first command, e.g. turn on lights",

"e": "command to execute (executable path etc.",

“p”: [ // array of parameters

“1”: {

“datatype”:””,

“min”:0,

“max”:100, //min value/length/size

“required”: true

}

]

},

"2": {

"name": "name of the second command ",

"e": “executable path"

},

}

}],

"aggregator": [{

"\_id": "59db76cdccc548be30e5249f",

"handler": "59bfdae2d7649a3070551ade",

"name": "Aggregator1",

"channel": "a/1", //channel to publish on

"description": "min/max and mean aggregator",

"poll": 18000, //run every 18 seconds

"deployment": "597f3056ef66be0648ef5bd3",

"active": true,

"\_\_v": 0,

"include\_raw": true, //include sensor message data in publications?

"topics": ["s/1/2/3/4/5/6/7/8/8/9/9/9/9", "s/4/5/22", "s/3/4/5"] //subscribe to these topics and aggregate their data

}, {

"\_id": "59db76cdccc548be30e5249a",

"handler": "59bfdae2d7649a3070551adb",

"name": "Aggregator2",

"channel": "a/2",

"description": "min/max and mean aggregator",

"poll": 60000,

"deployment": "597f3056ef66be0648ef5bd3",

"active": true,

"\_\_v": 0,

"topics": ["s/1/2/3/4/5/6/7/8/8/9/9/9/9", "s/4/5/22", "a/1/2/3/4/5"]

}],

"sensor": [{

"\_id": "59db98407e3a9fb060064ebf",

"id": "sn1234",

"name": "first real sensor",

"channel": "s/1/2/3/4/5/6/7/8/8/9/9/9/9", //publish data from this sensor on this topic

"description": "just generate a random number from minValue to minValue + range",

"handler": "59db908c327a46cb841ac004",

"poll": 60000,

"config": {

"minValue": 1234,

"range": 100

},

"location": "59bfd3ba9db46f37b858ed29",

"active": true,

"\_\_v": 0

}]

},

"mqttServers": [{ //what MQTT servers to connect or fail over to

"mqttServerIP": "127.0.0.1",

"mqttServerPort": "1883",

"priority": 1,

"\_id": "59df294d3a092a2b805b6570"

}, {

"mqttServerIP": "10.0.0.1",

"mqttServerPort": "1883",

"priority": 10,

"\_id": "59df294d3a092a2b805b656f"

}]

}

## Health Messages: H Topic

Health messages are designed to inform the platform of the current state of each device in terms of storage, memory, network and CPU utilisation. The purpose of these messages is to allow the Platform (or a user) balance roles across multiple devices in order to eliminate over or under utilisation of devices which could result in inefficiencies or bottlenecks across a deployment.

The platform sends an empty health message to a particular device on topic “H/<device path>”, where <device path> is the unique path to the relevant device. The device will use a range of Operating System commands or utilities (e.g. “top”, “df” etc.) to collate a set of statistics to return to the platform on topic “h/<device path>”

The structure of the health message has not been defined to date, and it may be the case that different types of data could be requested by the platform by tailoring the initial, “H”, message sent.

## Error Messages: E Topic

Error messages sent from the deployment have a different structure to most other types of messages in that the second and third elements define the role associated with the error and the type of error experienced. For example, a message topic “e/b/1/<device path>” contains an error of type 1, “no path to device”, relating to a broker on the device on path <device path>.

The following table outlines the error types and priority ranges for the deployment

|  |  |  |
| --- | --- | --- |
| Path | Error type | Description |
| e/a/1 | Aggregation error – data not received |  |
| e/a/2 | Aggregation error – data in incorrect format |  |
| e/a/3 | Aggregation error – received data outside of range |  |
|  |  |  |
| e/b/1 | Broker Error – no path to device | The device path has been omitted from the broker message |
|  |  |  |
| e/c/1 | Controller Error – invalid command id | The supplied does not match a command on the controller |
| e/c/2 | Controller Error – required parameter not supplied | One or more required parameters for the specified command was not supplied |
|  |  |  |
| e/r/1 | Coordinator Error- |  |
|  |  |  |
| e/m/1 | Commander Error - |  |
|  |  |  |
| e/d/1 | Device Error |  |
|  |  |  |
| e/s/1 | Sensor Error |  |

## Handler Messages: N Topic

Handler messages are used to push new or updated handler files from the Platform to a device. Handler messages are typically sent in advance of a configuration message from the Platform so that the required handlers are available prior to the configuration data being verified.

The structure of a handler message is as follows:

Topic: “N/<device path>”

Message: {

“id”: <the unique identifier for the handler>

“handler”: <base64 encoded representation of the handler file>

}

On receipt of the message, a device will need to decode the Base64 encoded handler file and save it with the name specified in “id” – this “id” filename can then be referenced in configuration messages (e.g. for Aggregator or Sensor Operations)

# Operational Messages

## Message brokering: B Topic

Messages from the Platform are encapsulated in broker messages before being published to the deployment. Broker messages do not have a lowercase variant as they will always travel away from the platform.

For example:

The platform wants to execute command number 1, on controller “1/2/3/4/5” with parameters “a” and 0. The greyed area below represents the corresponding message to be sent to the controller, while “p” is the destination for the message. (similarly, “C/5/4/3/2/1” could be used to send configuration data to device “5/4/3/2/1”)

The broker topic, “t”, is the hierarchy of brokers used to route the message.

Path: see items 1 – 5 below

Message: {

“t”: “B/5/V/6/4”,

“p”: “X/1/2/3/C/2”,

“m”: {

“c”:”1”,

“p”: [

“1”: ”a”,

“2”: “0”

]

}

}

1. coordinator publishes on "B/5"
2. Broker subscribed to “B/5, publishes same message on "B/5/V"
3. Broker subscribed to “B/5/V” publishes same message on “B/5/V/6”
4. Broker subscribed to “B/5/V/6”, publishes same message on "B/5/V/6/4”
5. Broker subscribed to “B/5/V/6/4”, publishes **included command message** on "X/1/2/3/C/2"
6. device subscribed to " X/1/2/3/C/2" executes the command

## Sensor Readings: S Topic

Sensor readings are sent on topic “s” – there is no upper case variant as the data will only ever travel towards the platform. Each message is sent on “s/<sensor path>” where <sensor path> is the path specified in the sensor configuration.

Each sensor topic is subscribed to by one or more Aggregators and the Aggregator handler file defines the expected structure of the sensor message and the operations to perform on it. In this way, HIOT can support any type of sensor and process any format of data – from simple numeric temperature or humidity values through to multi-dimensional data and even voice and video data. This allows HIOT to be deployed across a vast range of use cases and deployment types.

## Execution messages: X Topic

Execution messages are sent to a Controller in order to execute a command as specified in the Controller’s configuration.

In addition to the Platform, which can publish execution messages on topic “X”, Commanders within the deployment can also publish execution messages using the lower case variant, “x”. In either case, these message specific a controller, a command to execute and zero or more parameters for the command.

Unlike other message types, execution messages are encapsulated by Broker messages, and cannot be sent directly to a controller. The command element within an execution message is defined below

command:

{

"c":"the id of the command to execute,

"p":[

{"p1":"the first parameter"},

...

{"pn":"the nth parameter}

]

}

## Aggregation Results: A topic

- "a","aggregation output from an aggregator, can be sent via the coordinator, or as an input to another aggregator

{

"t":<timestamp>,

[

"1": first data element (e.g. average (mean) value)

"2": second data element (e.g. max)

"D": ["raw" data (from input sensors and/aggregators), if required]

]

}

## Event Messages: V Topic

- "V": platform notifies deployment/device of an event..??

- "v": device notifies platform of an event (e.g. temperature exceeds predefined limit)

"v" paths include a type as the second element and priority as the third element(e.g. "v/e/1" is a high priority error)

## Query Messages: Q Topic

- "Q": a query from the platform to the device specified in the remainder of the path - to support additional functionality

- "q": a query from a device - to support additional functionality

## Response Messages: R Topic

- "R": Response from the platform - platform has responded to a query from the deployment (requires "smart" roles)

- "r": device responds to the platform - e.g. confirm a role has been moved to a device

# Use cases

## Onboarding a device

## Data aggregation

### Excluding "raw" data

### Including "raw" data

## Command execution from platform

## Command execution from commander

## Updating "health" statistics

## "Balancing" a deployment

## Restoration in the event of a platform failure

Orphaned devices (e.g. where the platform data has become corrupted and the deployment is no longer defined) can recreate the relevant deployment based on their configuration file